



Embedded Linux  
Conference

Europe



OpenIoT Summit  
Europe

# Common attacks on IoT devices

**Why you can not win**

@binarychrysh



# Agenda

- What is IoT? And why is security important?
- Software attacks
- Hardware attacks
- Example attack stories
- Take-aways

# What is IoT?



# Approach

- *Analysis*: Inspect components, datasheets, firmware update process, contents of flash
- *Code execution*: Tamper with firmware update process, rewrite persistent memory content, gain access over debug channels/JTAG
- *Communication channel*: Get feedback from device over JTAG, serial console, etc

# Software

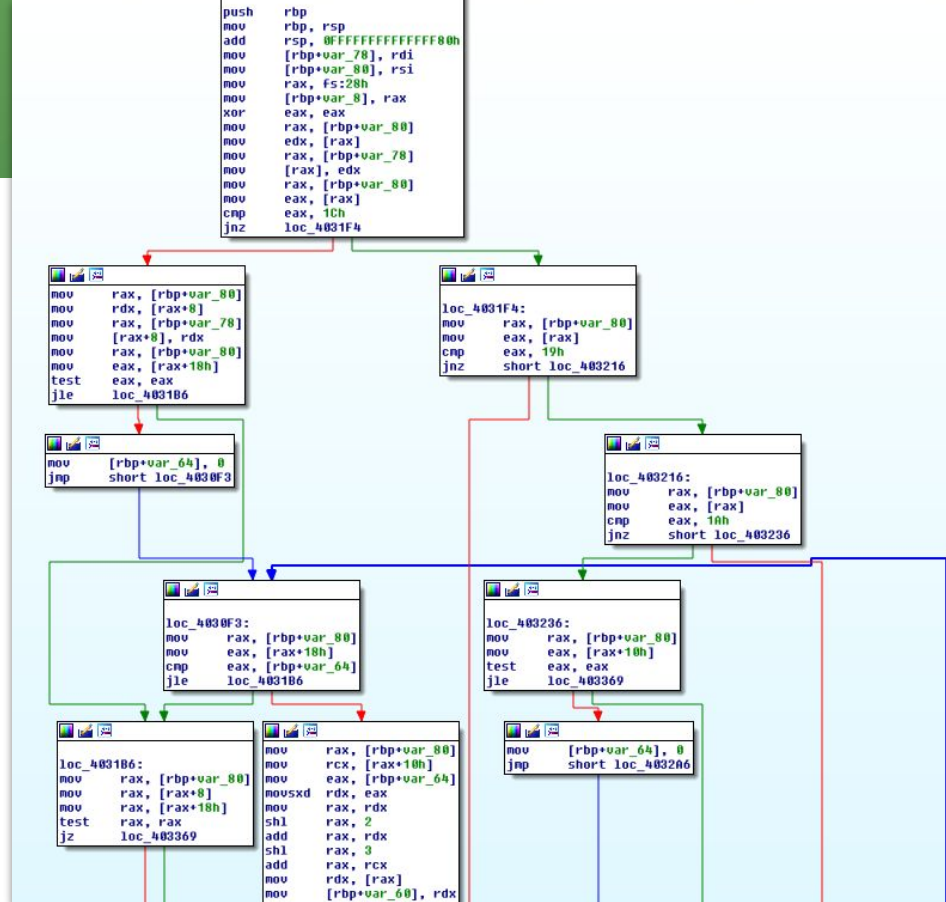
# Software

## Where to get the firmware

- Dump from device memory
- Download from manufacturer FTP server/search on ftp index sites
- Get from CD/DVD
- Wireshark traces of firmware updates

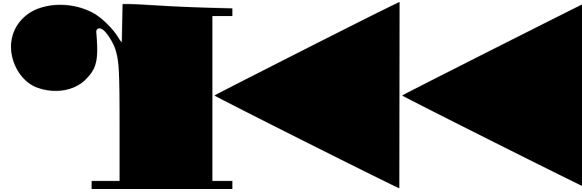
## Analyse firmware

- Understand file format from firmware update routine
- Search for code/string on code.google.com, sourceforge.net, ..
- Decompile, compile, tweak, fuzz
- If not stripped and human readable strings, it's easier to reverse



# Attacker Tools

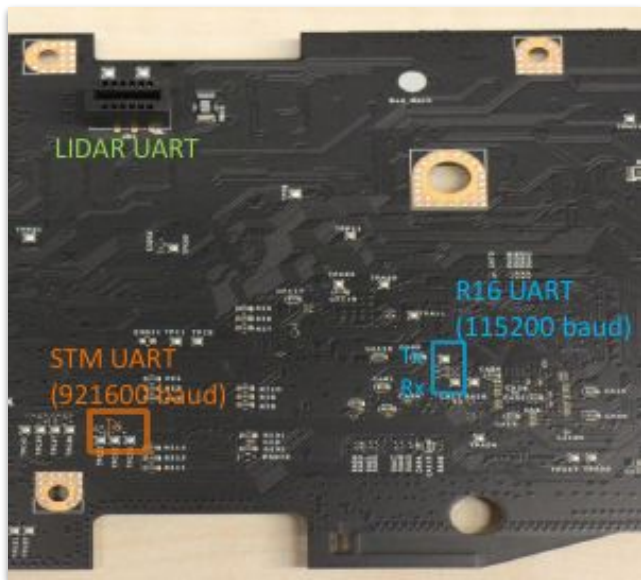
- **Software:**
  - **Binary reversing:**
    - IDA Pro
    - radare2
    - binaryninja
  - **Bug finder:**
    - Flawfinder
    - Metasploit Framework
  - **Firmware analysis:**
    - firmwalker (with binwalk, cpu\_rec)
    - firmware-analysis-toolkit
    - FACT (firmware analysis and comparison tool)
  - **Web testing:**
    - ZAP, sqlmap, sslyze, Gobuster (see OWASP)
  - **Debugging:**
    - GDB & OpenOCD





# Hardware

# Hardware



[Backside layout mainboard Xlaomi Vacuum Cleaner robot by *Dennis Giese and Daniel Wegemer*]

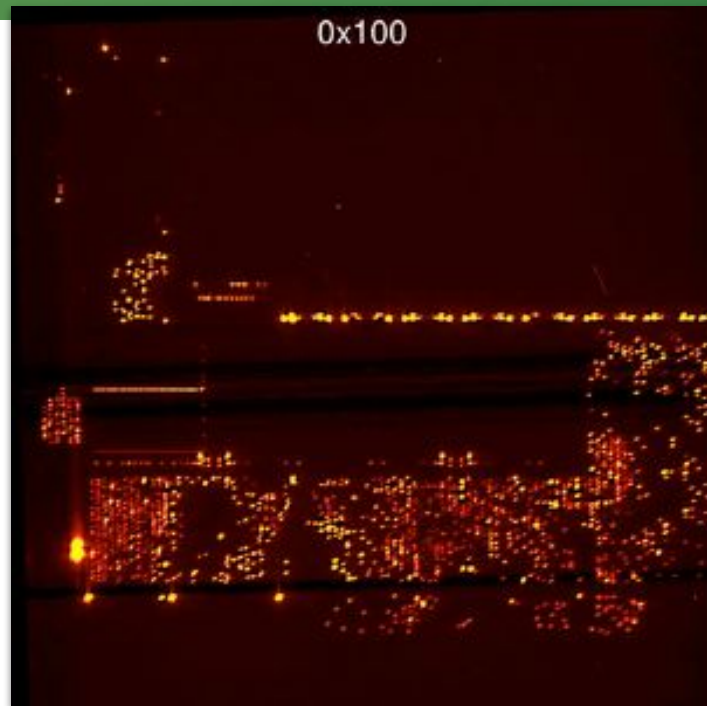
## Non-invasive attacks

- Search for UART, JTAG, etc
- Write protection security fuses not enabled => Patch bootloader
- Hardware Fuzzing (automatically send random data and monitor whether device crashes)
- **Side channel attacks**
  - **Timing attacks**
    - Computation time depends on value of secret data
    - Cache miss and cache hit have huge timing difference => find access pattern in dependence of timing difference

# Hardware

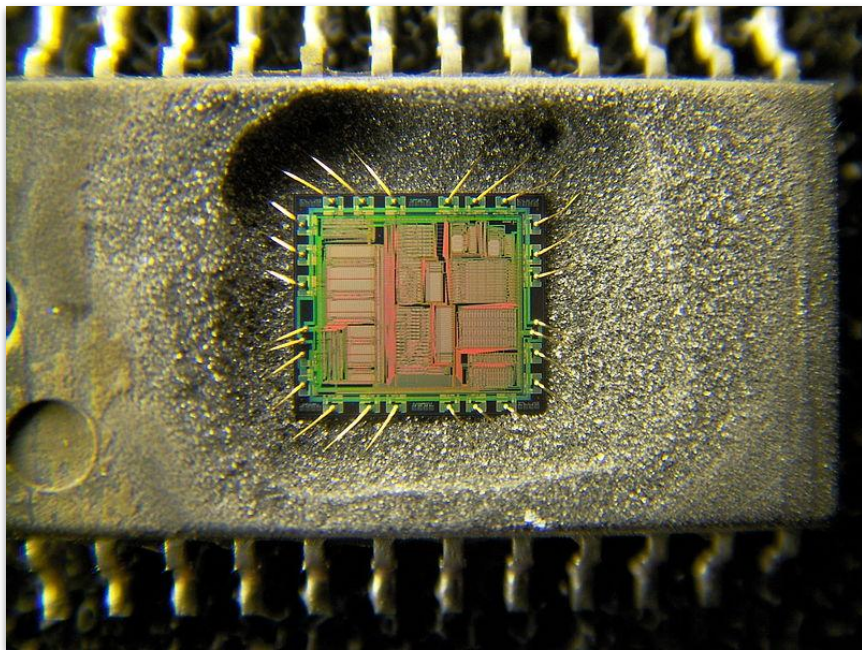
## Non-invasive attacks

- **Side Channel Attacks (2)**
  - **Hardware Glitching**
    - very high/low voltage
    - alter clock period during execution
  - **Power Analysis**
    - Power consumption of a chip depends on the secret data that is computed on the chip):
      - SPA (Simple power analysis)
      - DPA (Differential power analysis)
    - EM Radiation channel
    - Acoustic channel



[Visible and infrared light emitted by switching transistors/ by *Dmitry Nedospasov*]

# Hardware



[Yamaha audio IC decapsulated by Olli Niemitalo/ CC0 1.0]

## Semi-invasive attacks

- Decapping package
- Infrared light/photon emission analysis of backside to find location for attack
- Then use laser to flip bits and break crypto

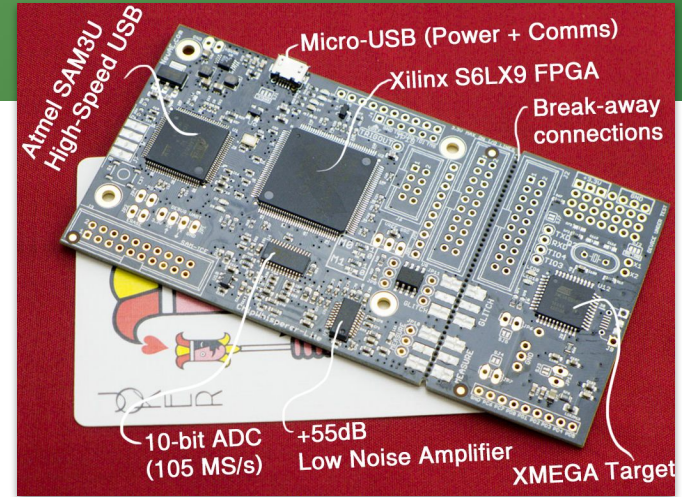
## Fully-invasive attacks

- Much effort, but 100% success rate
- Modify chip with FIB (Focused Ion Beam)
- Microprobing
- Linear code extraction (LCE)

# Attacker Tools

- **Hardware:**

- Oscilloscope
- Logic Analyzer (e.g. Saleae)
- JTAG:
  - GoodFET, BusBlaster, BusPirate, JTAGulator, JTAGenum, Black Magic Probe
- Side Channel Attacks:
  - ChipWhisperer (power analysis, glitching attacks)
- USB:
  - Facedancer
- SDR:
  - HackRF



ChipWhisperer



BusPirate



HackRF

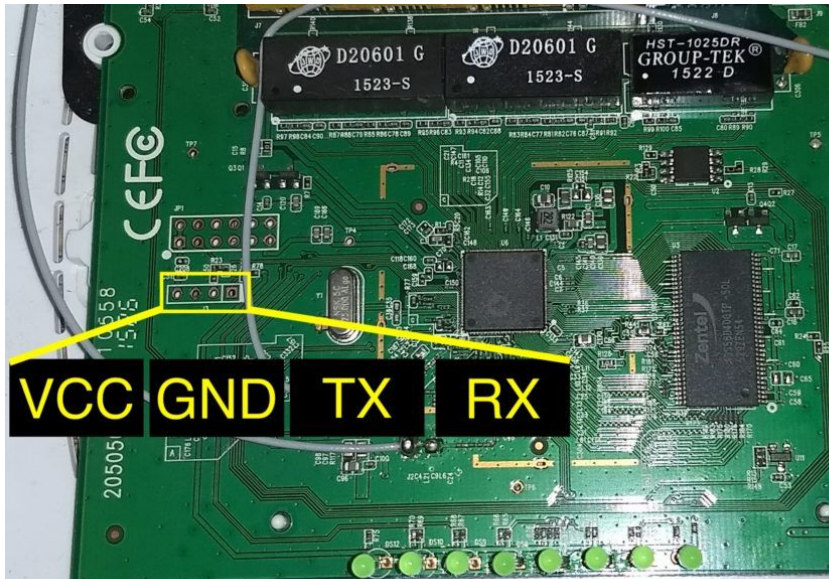


# Real world attacks

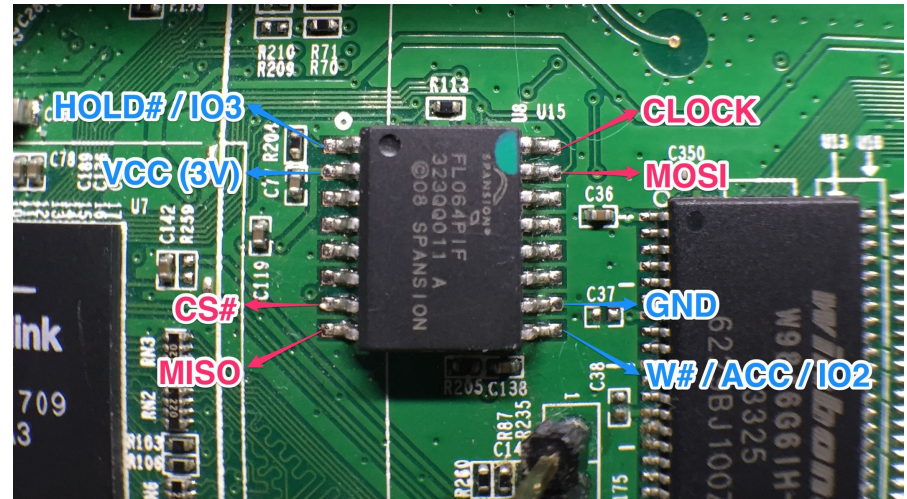


# Real world attacks\*

- **UART** (populated or not): Usually device boots into special console/root console



[From 5-Min Tutorial: Gaining Root via UART by @konukoli]



[From Hack The World by Juan Carlos Jiménez]

# Real world attacks\*

- **Root with U-Boot:**
  - Access bootloader shell, add **init=/bin/sh** into kernel cmdline
  - Will execute **preconfigured script name** 'xyz' => replace script with own script
  - **Short pins** on NAND, power on => boot into corrupted U-Boot environment
- **Hardcoded/base64 encoded username and password** in binary
- **Bruteforce** easy password



[BGA by Smial / GFDL-1.2]

```
la      $t9, _strcpy
lui     $s1, 0x5A
move    $a0, $s0
jalr    $t9, _strcpy
addiu   $a1, $s1, (aAdmin - 0x5A0000) # "admin"
lw      $gp, 0x60+var_50($sp)
addiu   $a1, $s1, (aAdmin - 0x5A0000) # "admin"
la      $t9, _strcat
nop
jalr    $t9, _strcat
move    $a0, $s0
lw      $gp, 0x60+var_50($sp)
nop
la      $t9, _strlen
nop
jalr    $t9, _strlen
move    $a0, $s0
move    $a2, $v0
addiu   $a0, $sp, 0x60+var_48
jal     md5checksum
move    $a1, $s0
lw      $v0, 0x60+var_44($sp)
lui     $a0, 0x5E
addiu   $v1, $a0, (dword_5E3E00 - 0x5E0000)
sw      $v0, (byte_5E3E04 - 0x5E3E00)($v1)
li      $v0, 1
sw      $v0, (dword_60D9C8 - 0x610000)($s2)
lw      $v0, 0x60+var_48($sp)
lw      $gp, 0x60+var_50($sp)
sw      $v0, dword_5E3E00
```

[Reverse Engineering the TP-Link HS110 by Lubomir  
Stroetmann, Consultant and Tobias Esser, Consultant/  
© Softscheck]



\* See Talk "Hack All The Things: 20 Devices in 45 Minutes" by gtvhacker



# Real world attacks\*

**TFTP Firmware Download from PC to Camera**

No.	Time	Source	Destination	Protocol	Length	Info
386	389.302758	Dell_02:19:5e	Hangzhou_4d:01:47	ARP	42	Who has 192.168.1.64? Tell 192.168.1.128
387	389.303143	Hangzhou_4d:01:47	Dell_02:19:5e	ARP	60	192.168.1.64 is at 44:19:b6:4d:01:47
388	391.268395	Hangzhou_4d:01:47	Broadcast	ARP	60	Who has 192.168.1.128? Tell 192.168.1.64
389	391.268435	Dell_02:19:5e	Hangzhou_4d:01:47	ARP	42	192.168.1.128 is at 34:c6:47:02:19:5e
390	391.268455	Hangzhou_4d:01:47	Broadcast	ARP	60	Who has 192.168.1.128? Tell 192.168.1.64
391	391.268468	Dell_02:19:5e	Hangzhou_4d:01:47	ARP	42	192.168.1.128 is at 34:c6:47:02:19:5e
392	391.268513	192.168.1.64	192.168.1.128	TFTP	84	Read Request, File: digicap.dav, Transfer type: octet, timeout=5, blksize=512
393	391.275663	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 1
394	391.276482	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 1
395	391.276586	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 2
396	391.277613	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 2
397	391.277691	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 3
398	391.278416	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 3
399	391.278486	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 4
400	391.279230	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 4
401	391.279296	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 5
402	391.280028	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 5
403	391.280089	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 6
404	391.281167	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 6
405	391.281229	192.168.1.128	192.168.1.64	TFTP	558	Data Packet, Block: 7
406	391.281982	192.168.1.64	192.168.1.128	TFTP	60	Acknowledgement, Block: 7

> Frame 1: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0  
> Ethernet II, Src: Hangzhou\_4d:01:47 (44:19:b6:4d:01:47), Dst: Broadcast (ff:ff:ff:ff:ff:ff)  
v Address Resolution Protocol (request)  
  Hardware type: Ethernet (1)  
  Protocol type: IPv4 (0x0800)  
  Hardware size: 6  
  Protocol size: 4  
  Opcode: request (1)  
  Sender MAC address: Hangzhou\_4d:01:47 (44:19:b6:4d:01:47)  
  Sender IP address: 192.168.1.64  
  Target MAC address: 00:00:00:00:00:00 (00:00:00:00:00:00)  
  Target IP address: 192.168.1.128

0000 ff ff ff ff ff ff 42 19 b6 4d 01 47 08 06 00 01 .....D..M.G...  
0010 00 00 06 04 00 01 44 19 b6 4d 01 47 00 00 01 45 .....D..M.C...  
0020 00 00 00 00 00 00 c0 a8 01 80 53 57 4b 48 00 00 .....SHKH..  
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....  
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

- Write su binary into eMMC fs
- **Command injection**
  - system("ls %s"): will reboot on user input "; reboot;")
  - often in WEP or Wifi password field of Configuration Web page, Network folder names, ..
  - In URL parameters (http://foobar/subpage?action=command&command=reboot)
- App installation /Firmware update over **unencrypted HTTP/FTP** => can be intercepted
- SMB share without restrictions, run su binary via adb

[How to Fix a Bricked Hikvision IP Camera Firmware by Bob Jackson]

\* See Talk "Hack All The Things: 20 Devices in 45 Minutes" by gtvhacker

# Real world attacks: Xiaomi Vacuum Cleaning Robot\*

**Micro USB Port:** was authentication protected



**Serial communication:** Didn't find



**Port Scan:** No suspicious open ports



**Sniff network traffic**



**Recovery mode:** Shorting BGA pins with aluminium foil



[CC by 4.0 34C3 media.ccc.de]

# Real attack stories: PLC\*

- Downgrading to older firmware
- Physical mapping of JTAG not easy to find
- Injecting code into firmware update



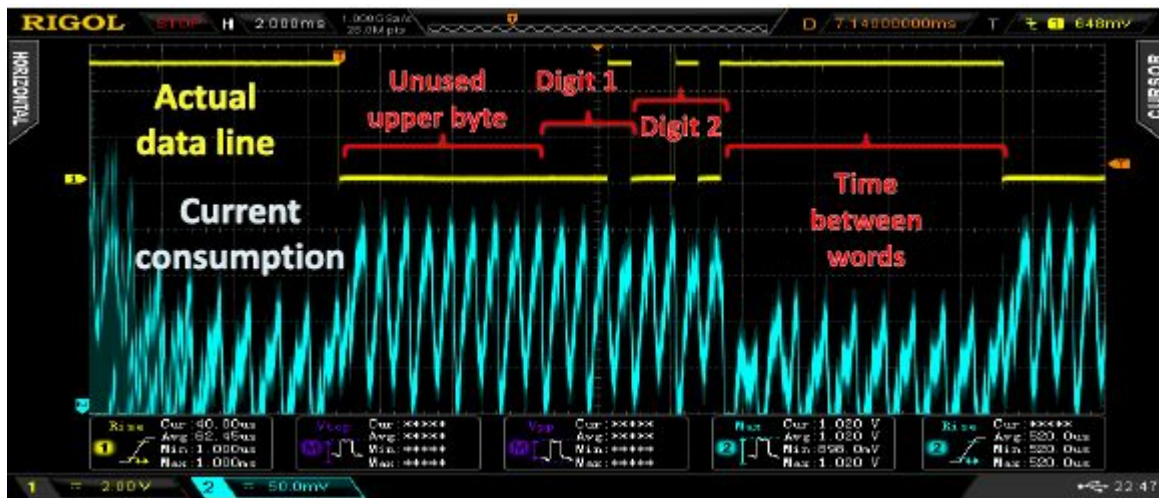
- Injecting code via flash reprogramming
  - rewrote bootloader after partly desoldering pins asserting write protection
  - MitM like setup for quick prototyping and testing of bootloader replacement code



Picture:  
<https://www.astiautomation.ro/en/product/plc-canopen-training-panel-s7-1200-siemens/>

# Real attack stories: Electronic Safe Lock\*

- Resistor in series to battery and lock
- Amplified current => Power analysis Side channel attack (high current consumption => 0 read from EEPROM, low current => 1 read from EEPROM)
- *Mitigate*: Don't store secret in EEPROM



Sargent & Greenleaf 6120-332  
[by Plore]

[by Plore]

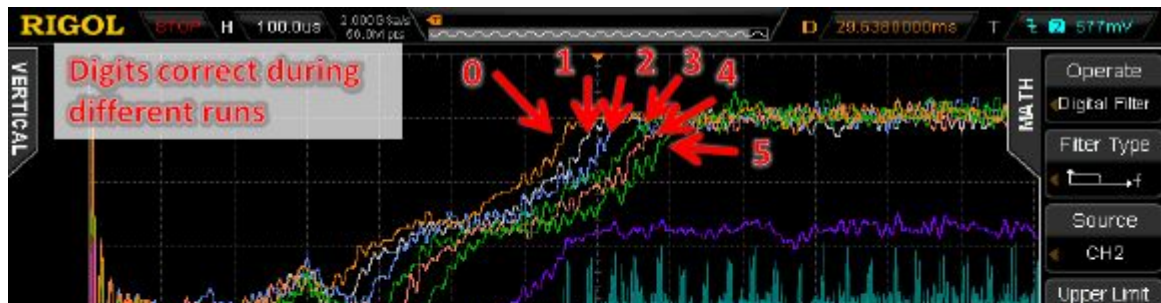
\* See Talk "DEF CON 24 - Plore - Side channel attacks on high security electronic safe locks" by Plore

# Real attack stories: Electronic Safe Lock\*

- Timing attack: The correct key will have a longer delay
- Problem: 5 tries, then locked out for 10 minutes
- Counter of tries stored in EEPROM
- Reset counter by turning off MCU shortly after write of counter started, where cell is erased but not written yet
- Mitigate: Constant time for comparison, hashed secrets



S&G Titan PivotBolt [by Plore]



\* See Talk “DEF CON 24 - Plore - Side channel attacks on high security electronic safe locks” by Plore

# Protection

# Protection\*

- **Buffer/Stack Overflow Protection, heap overflow protection**
  - Use **safe equivalent** functions (gets()->fgets())
  - Verify buffer bounds
  - Secure compiler flags (-fPIE, -fstack-protector-all, -Wl,-z,noexecstack, -Wl,-z,noexecheap,...)
  - See [https://wiki.debian.org/Hardening#Using\\_Hardening\\_Options](https://wiki.debian.org/Hardening#Using_Hardening_Options)
- **Injection (SQL/command injection, XSS) protection for webserver**
  - Whitelist commands
  - No user data into OS system commands
  - Validate input & output



# Protection\*

- **Firmware Updates with cryptographic signatures, update over TLS**
  - Force updates for high critical bugs
  - Anti-rollback protection
  - Infrastructure with pub-priv key for verifying signed packages
  - **Don't Roll Your Own Crypto!**
- **Secure sensitive information**
  - **No hardcoded** secrets (usernames, passwords, tokens, priv keys,.).
  - Store secrets only in protected storage (NOT EEPROM, flash)
  - Use Trusted Execution Environment (TEE) or security element (SE), TrustZone (for ARM)



# Protection\*

- **Identity Management**

- Separate accounts for internal/remote web management, internal/remote console access
- No sessionIDs/Tokens/Cookies in URL (can be replayed)
- Tokens should be randomized, and invalidated on logout
- Secure and complex password for accessing UART, EEPROM, ssh
- Each device: individual secret (one device's gets hacked, the others stay safe)

- **Hardened toolchains, libraries and frameworks**

- **Remove unused** language/shell interpreters (/bin/dash, /bin/bash, /bin/ash, /bin/zsh, ..), dead (debugging) code (dead code which can be used for attacks), unused libs
- **Disable ancient legacy** protocols (ftp, telnet, ..)
- Remove debugging interfaces
- Remove (or secure) ~~backdoors~~ management interfaces for consumer support/debugging purposes,...usually with root privilege
- Check third party code and SDKs

# Protection\*

- **Keep kernel, frameworks & libraries up to date**
  - Use package managers opkg, ipkg
  - Check against vulnerabilities DBs
  - Load tools to check third party code and components (retirejs, libscanner, nsp, lynis, owasp zap, ..),
- **Threat modeling**

# Take-aways

- Main attack vectors: web-interface, crypto, outdated/unpatched firmware, sniffing unencrypted communication and cleartext passwords..
- Don't have your key or password fixed in your binary, store secrets in hardware protected place
- Integrate security tests into your CI/development cycles
- There is always a way to hack a system, just a matter of cost and time

# Questions?



# Ressources

- [https://www.owasp.org/index.php/OWASP\\_Embedded\\_Application\\_Security](https://www.owasp.org/index.php/OWASP_Embedded_Application_Security)
- [http://www.sharcs-project.eu/m/documents/papers/a01-cojocar.p df](http://www.sharcs-project.eu/m/documents/papers/a01-cojocar.pdf) (Off-the-shelf Embedded Devices as Platforms for Security Research)
- <https://www.handymanhowto.com/how-to-fix-a-bricked-hikvision-ip-camera-firmware/>
- <http://jcjc-dev.com/2016/06/08/reversing-huawei-4-dumping-flash/>
- <http://konukoi.com/blog/2018/02/16/5-min-tutorial-root-via-uart/>

# Recommended Talks

- “34C3 - Unleash your smart-home devices: Vacuum Cleaning Robot Hacking” by Dennis Giese, Daniel Wegemer from TU Darmstadt
- “Hardware Hacking - Extracting Information From Chips“ by Dmitry Nedospasov
- “Lockpicking in the IoT...or why adding BTLE to a device sometimes isn't smart at all“ by Ray
- “DEF CON 24 - Plore - Side channel attacks on high security electronic safe locks” by Plore
- Hack All The Things: 20 Devices in 45 Minutes
- “Black Hat 2013 - Exploiting Network Surveillance Cameras Like a Hollywood Hacker” by Craig Heffner



**Embedded Linux  
Conference**

Europe

---



**OpenIoT Summit**  
Europe